探索 Iris（鸢尾花）数据集

CS245　数据科学基础　陆朝俊

1　问题描述

统计学主要从数量方面研究事物，目的是探索事物（一般为数据集）的数量特征。统计学中的描述统计学借助图表或概括性的数值将数据集展示为清晰可理解的形式。在之前的研究中，我已经使用了图表对 Adults 数据集进行了探索和可视化展示，这次研究的主要目标是探索 Iris 数据集，并通过一些概括性的数值对其进行展示，详细目标如下：

1. 探索 Iris 数据集的基本属性（如数据集总体描述、数据维数、特征名称等）；
2. 探索各特征的最小值、最大值、均值、中位数、标准差；
3. 探索各特征之间，以及特征与目标之间的相关性（相关系数）。

2　解决方案[1]

2.1　数据集及其基本属性的获取

Iris 数据集也称鸢尾花卉数据集，是一个经典且常用的多重变量分析数据集，已集成在 Python 的 scikit-learn 模块下，无需另外下载。使用 pip 或其余包管理器安装好 scikit-learn 模块后，只需在命令行输入以下命令即可读取 Iris 数据集：

```
1 from sklearn import datasets
2 iris = datasets.load_iris()
```

数据集获取完毕后，只需要调用数据集对象 iris 的一系列方法即可获得其基本属性（见下方示例代码），详细结果可参见 3.1 小节。

```
1 print(iris.DESCR)
2 print(iris.target)
3 print(iris.data.shape)
4 print(iris.feature_names)
5 print(iris.target_names)
```

2.2　各特征的分析

通过第一步的分析，我已经确定 Iris 所有观测实例的特征数据为 numpy.ndarray 类型，而对于数值分析（如求平均值、方差等），numpy 模块已经提供了简便且高效的函数，因此这一部分的分析可以分为三步：

1. 从 Iris 对象中提取出所有实例的某个特征；
2. 使用 numpy 模块中的函数求得该特征的最小值、最大值、均值等；
3. 展示求得的结果，并据此做进一步的分析。

[1]本次作业的所有代码实现可参见附录 A.2

2.3 相关性分析

这一部分的分析方法和 2.2 小节基本相同，仍然是使用 numpy 模块中的函数求得各特征之间、以及特征与目标之间的相关系数，从而分析其相关性。分析中我采用一般的相关系数判别标准，即取绝对值后，0-0.09 为没有相关性，0.1-0.3 为弱相关，0.3-0.5 为中等相关，0.5-1.0 为强相关。分析的详细结果可参见 3.3 小节。

3 结果展示

3.1 Iris 数据集基本属性

使用 2.1 小节里描述的方法，可以获得 Iris 数据集的所有重要属性。在这次探索中，我主要获取了 Iris 中与数据类型、结构以及名称相关的属性（如实例的数据维数、类别名称等），具体结果可参见下表（Iris 数据集官方提供的基本描述可参见附录 A.1.1）。

表 1: Iris 数据集基本属性

属性	值
作者	R.A. Fisher
创建时间	1988.7
实例的数据个数	150
实例的数据类型	numpy.ndarray
实例的数据维数	(150, 4)
类别的数据类型	numpy.ndarray
实例的类别值	0,1,2
实例的类别维数	(150,)
每类的实例个数	50
特征数值详情	正实数，单位为 cm
特征名	萼片长度，萼片宽度，花瓣长度，花瓣宽度
类别名称	setosa（山鸢尾），versicolor（杂色鸢尾），virginica（维吉尼亚鸢尾）

结果显示，Iris 数据集包含 150 个鸢尾花实例，每个实例拥有 4 个特征，分别表示萼片长度，萼片宽度，花瓣长度以及花瓣宽度（单位均为 cm），按照 numpy.ndarray 的格式存储；分类目标有 3 类，为 setosa（山鸢尾），versicolor（杂色鸢尾）和 virginica（维吉尼亚鸢尾），分别使用 0，1，2 表示。

由此可知，Iris 属于规模极小、数据格式简单的多重变量分析数据集，是实验简单分类器算法以及机器学习入门实践的良好选择。了解数据集的基本属性和数据结构，可以方便之后的探索和分析。

3.2 各特征的数值描述

这一小节展示了 Iris 数据集中各特征的数值描述。除计算基本的最小值，最大值，均值，中位数，标准差外，我还通过 python 的 stats 模块计算了各特征的上、下四分位数，从而更好地了解数据分布。详细结果可参见表 2 （考虑到有些统计量无单位，因此在表格中不把单位显式地表示出，各特征单位均为 cm）。

表 2: Iris 数据集各特征的数值描述

	萼片长度	萼片宽度	花瓣长度	花瓣宽度
最小值	4.3	2.0	1.0	0.1
最大值	7.9	4.4	6.9	2.5
均值	5.843	3.054	3.759	1.120
中位数	5.80	3.00	4.35	1.30
标准差	0.825	0.432	1.759	0.761
方差	0.681	0.187	3.092	0.579
极差	3.6	2.4	5.9	2.4
下四分位数	5.1	2.8	1.6	0.3
上四分位数	6.4	3.3	5.1	1.8

从结果来看，萼片宽度这一特征有着最小的方差，分布最为紧密，而花瓣长度的分布最为稀疏。其余数值属性（如最大值、均值等）更多的是反映该特征本身的性质，不适用于对比性的分析，因此仅将其列在表格中，不做进一步的对比分析。

3.3 各特征及特征与目标之间的相关性

这一小节展示了对 Iris 数据集中相关性的探索结果。我将各特征之间及特征与目标之间的相关系数组成表格（见表 3），由于相关系数与两变量的排列顺序无关，即 $corrcoef(X,Y) = corrcoef(Y,X)$，因此表格呈现类似于对称矩阵的分布。

表 3: Iris 数据集各特征及特征与目标之间的相关性

	萼片长度	萼片宽度	花瓣长度	花瓣宽度
萼片长度	1.000	-0.109	0.872	0.818
萼片宽度	-0.109	1.000	-0.421	0.357
花瓣长度	0.872	-0.421	1.000	0.963
花瓣宽度	0.818	-0.357	0.963	1.000
目标	0.783	-0.420	0.949	0.956

由上表可知，花瓣长度与花瓣宽度这两个特征和鸢尾花种类有极强的相关性；萼片长度与鸢尾花种类的相关性较之稍弱，但也属于强相关；而萼片宽度与鸢尾花种类间仅有中等相关性。

根据各特征之间的相关系数分析，相关性较强的特征有：萼片长度-花瓣长度，萼片长度-花瓣宽度，花瓣长度-花瓣宽度；相关性较弱的特征有：萼片长度-萼片宽度。

A 附录

A.1 Iris 数据集的其余基本属性

A.1.1 Iris 数据集的基本描述

Iris 数据集基本描述

```
Notes
-----
Data Set Characteristics:
    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the
       class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20  0.76     0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML iris datasets.
http://archive.ics.uci.edu/ml/datasets/Iris

The famous Iris database, first used by Sir R.A Fisher

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the
   field and
is referenced frequently to this day.  (See Duda & Hart, for example.)
     The
data set contains 3 classes of 50 instances each, where each class
```

```
   refers to a
type of iris plant.  One class is linearly separable from the other 2;
    the
latter are NOT linearly separable from each other.

References
----------
   - Fisher,R.A. "The use of multiple measurements in taxonomic
     problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "
        Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda,R.O., & Hart,P.E. (1973) Pattern Classification and Scene
     Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New
     System
     Structure and Classification Rule for Recognition in Partially
        Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE
     Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s
     AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

A.2 主要代码

Iris 数据集探索代码

```python
from sklearn import datasets
import stats
import numpy as np
import xlwt

class IrisAnalyzer(object):
    def __init__(self):
        # iris data
        self.iris = datasets.load_iris()
        self.data = self.iris.data
        self.target = self.iris.target
        print(self.iris.DESCR)

        # save results as excel tables
        self.wbk = xlwt.Workbook()
        self.char_sheet = self.wbk.add_sheet('各特征分析')
        self.corr_sheet = self.wbk.add_sheet('相关分析')
```

```
18          self.metrics = {'max': np.max, 'min': np.min, 'avg': np.mean,
19                          'median': np.median, 'ptp': np.ptp,
20                          'std': np.std, 'var': np.var, 'q1': stats.quantile
                                ,
21                          'q3': stats.quantile}
22
23          # initialize the excel tables
24          for i, name in enumerate(self.iris.feature_names):
25              self.char_sheet.write(0, i + 1, name)
26          for i, m in enumerate(self.metrics.keys()):
27              self.char_sheet.write(i + 1, 0, m)
28          for i, name in enumerate(self.iris.feature_names):
29              self.corr_sheet.write(0, i + 1, name)
30          for i, m in enumerate(self.iris.feature_names):
31              self.corr_sheet.write(i + 1, 0, m)
32          self.corr_sheet.write(i + 2, 0, 'target')
33
34          print('----------analyzer started----------')
35
36      def run(self):
37          # starting to analyze Iris dataset
38          self.basic_att()
39          self.char_analysis()
40          self.corr_analysis()
41
42          self.wbk.save('results.xls')
43          print('----------results saved----------')
44
45      def basic_att(self):
46          # get basic attributes of Iris dataset
47          with open('basic_att.txt', 'w') as f:
48              f.write(self.iris.DESCR + '\n\n')
49
50              f.write('Type of data: ' + str(type(self.data)) + '\n\n')
51              f.write('Shape of data: ' + str(self.data.shape) + '\n\n')
52              f.write('Feature names: ' + str(self.iris.feature_names) +
                        '\n\n')
53
54              f.write('Target: ' + str(self.target) + '\n\n')
55              f.write('Type of target: ' + str(type(self.target)) + '\n\
                    n')
56              f.write('Shape of target: ' + str(self.target.shape) + '\n
                    \n')
57              f.write('Target names: ' + str(self.iris.target_names) + '
                    \n\n')
58
59      def char_analysis(self):
60          # analyzing each of characters
61          for i in range(self.data.shape[1]):
62              char_data = self.data[:,i]
```

```
63              for j, m in enumerate(self.metrics.keys()):
64                  if not m in ['q1', 'q3']:
65                      self.char_sheet.write(j + 1, i + 1, self.metrics[m
                            ](char_data))
66                  elif m == 'q1':
67                      self.char_sheet.write(j + 1, i + 1, self.metrics[m
                            ](char_data, p=0.25))
68                  elif m == 'q3':
69                      self.char_sheet.write(j + 1, i + 1, self.metrics[m
                            ](char_data, p=0.75))
70
71      def corr_analysis(self):
72          # analyzing the correlationships
73          for i in range(self.data.shape[1]):
74              char_data = self.data[:, i]
75
76              # corr among characters
77              for j in range(self.data.shape[1]):
78                  self.corr_sheet.write(j + 1, i + 1, np.corrcoef(
                        char_data, self.data[:, j])[0][1])
79
80              # corr between characters and target
81              self.corr_sheet.write(j + 2, i + 1, np.corrcoef(char_data,
                    self.target)[0][1])
82
83  def debug():
84      # just used to debug, ignore it
85      pass
86
87  if __name__ == '__main__':
88      ia = IrisAnalyzer()
89      ia.run()
```